25. (Amended) The computer program product as claimed in claim 23, further comprising changing the status of the object to global when the monitoring step determines that the object is accessible from either of a global root or other global object.

26. (Amended) The computer program product as claimed in claim 23 wherein the status of an object in the thread heap is changed to global if the object is assigned to a static variable or if the object is assigned to a field in a global object.

REMARKS/ARGUMENTS

This communication is responsive to a Final Office Action dated January 28, 2003 rejecting claims 1-26 of the instant patent application (hereafter the "Final Office Action"). Therefore, claims 1, 6, 14, 18, and 22 have been amended. Reconsideration of the application is respectfully requested.

I. Claim Rejections – Double Patenting

The Final Office Action provisionally rejected claim 1 as being unpatentable over claim 1 of co-pending patent application 09/337,824 under the doctrine of obviousness-type double patenting. Without conceding the contention of the Final Office Action, Applicant is hereby filing a Terminal Disclaimer obviating the rejection.

II. Claim Rejections – 35 USC §103(a)

The Final Office Action has introduced a new rejection of all twenty-six claims pending in the above-referenced patent application. The claims have now been rejected as being obvious in view of the combination of U.S. Patent 6,304,949 to Houlsdworth (hereafter, "Houlsdworth") and U.S. Patent 6,253,215 to Agesen (hereafter, "Agesen").

Houlsdworth relates to a data processing apparatus for handling multi-thread programs. The Examiner concedes that Houlsdworth does not discuss the claimed monitoring of each object to determine whether the object is accessible from any thread other than the given thread. However, the Examiner concluded that the Agesen patent makes up for this deficiency of the discussion in Houlsdworth and cites:

"To be effective, garbage collection techniques should be able to, first, identify references that are directly accessible to the executing program, and, second, given the reference to an object, identify references contained within that object, thereby allowing the garbage collector to transitively trace chains of references. Unfortunately, many of the described techniques for garbage collection have specific requirements which cause implementation problems, particularly when a garbage collector is charged with managing memory for a system having programs written in both native and target code. For example, the Java VM's garbage collector manages resources for Java code with relative ease; however, it requires additional facilities to manage resources for other native code and even then the garbage collector has significant limitations." Agesen, Col. 3, lines 19 *et seq.*

"Objects in the heap can be accessed by means of direct and indirect pointers. A direct pointer is the same as a reference in a local or global variable and can be used in a program to access an object. In contrast, an indirect pointer is a pointer to a direct pointer. Consistent regions of program code use only indirect pointers to reference objects. To access an object, for example, to write a value in a field of the object, the indirect pointer is "dereferenced," obtaining access to the direct pointer and thus access to the object itself. Regions of program code during which objects are accessed by using direct pointers are "inconsistent" regions because the dereferencing of an indirect pointer may copy a direct pointer value into a location not known by the garbage collector to contain such a pointer. Thus, garbage collection is not permitted during inconsistent regions of program code because it is not possible to determine exactly which slots in the stack

frame are pointers to objects in the heap. If the garbage collector relocates an object (as is often the case with a compacting garbage collector, for example), the collector may fail to update direct pointers that were obtained by dereferencing indirect pointers to the new location of the relocated objects." Agesen, Col. 11, lines 37 *et seq.*

The Examiner concluded that the combination of these references was motivated because it would prevent deleting the wrong data object to keep data consistency in a system. He cited no support for this conclusion.

Claim 1 as amended requires *inter alia*, for a given thread, monitoring each object in its heap, to determine whether the object is accessible by any thread other than the given thread. Applicant therefore respectfully traverses the rejection of claims 1-26 under 35 USC §103(a).

The US Patent and Trademark Office cannot simply reach conclusions based on its own understanding or experience -- or on its assessment of what would be basic knowledge or common sense. Rather, the Board must point to some concrete evidence in the record in support of these findings. See In re Zurko, 258 F.3d 1379 (Fed. Cir. 2001). In In re Dembizak, 175 F.3d 994, 998, 50 USPQ2d 1614, 1616 (Fed. Cir. 1999), the United States Court of Appeals for the Federal Circuit clearly restated the standard:

"We have noted that evidence of a suggestion, teaching, or motivation to combine may flow from the prior art references themselves, the knowledge of one of ordinary skill in the art, or, in some cases, from the nature of the problem to be solved, see Pro-Mold & Tool Co. v. Great Lakes Plastics, Inc., 75 F.3d 1568, 1573, 37 USPQ2d 1626, 1630 (Fed. Cir. 1996), Para-Ordinance Mfg. v. SGS Imports Intern., Inc., 73 F.3d 1085, 1088, 37 USPQ2d 1237, 1240 (Fed. Cir. 1995), although "the suggestion more often comes from the teachings of the pertinent references," Rouffet, 149 F.3d at 1355, 47 USPQ2d at 1456. The range of sources available,

however, does not diminish the requirement for actual evidence. That is, the showing must be clear and particular."

In the instant case the Final Office Action cites no evidence that either of the cited patents suggests the claimed invention and moreover there is no teaching of any suggestion, motivation, or rationale of the desirability of the advantage of the claimed invention cited by the Final Office Action (i.e., preventing deleting the wrong data object to keep data consistency in a system).

In addition, there is no evidence whatsoever in either of the cited patents of any discussion of thread-local objects (i.e., those accessible only from a given thread) or of any potential advantages there may be in identifying them. The claimed invention identifies, for each thread, the set of objects which are accessible to ONLY that thread. By so doing the objects can be garbage collected from those threads without affecting the other threads in the system. This is a concept not mentioned by prior art and certainly not by Ageson or Houlsdworth. So, when Ageson talks about 'identify references that are directly accessible to the executing program' it is specifically identifying references from all the threads in that program and thus not covering what is claimed (identifying accessibility from a given thread only).

Claims 2-17 depend upon claim 1 and are patentable over the cited patents for at least the same reasons that claim 1 is patentable. Claim 18 is a system counterpart of the method of claim 1 and is patentable over the cited patents for at least the same reasons that claim 1 is patentable. Claims 19-21 depend upon claim 18 and are patentable over the cited patents for at least the same reasons that claim 1 is patentable. Claim 22 is a program product counterpart of the method of claim 1 and is patentable over the cited patents for at least the same reasons that claim 1 is patentable. Claims 23-26 depend upon claim 22 and are patentable over the cited patents for at least the same reasons that claim 22 is patentable.

Therefore, the Applicant respectfully submits that the claims presented are patentable over the patents cited.

Attached hereto is a marked-up version of the changes made to the specifications and claims by the current amendment. The attached page is captioned "**Version with markings to show changes made.**"

Applicant respectfully requests that a timely Notice of Allowance be issued in this case.

Respectfully submitted,

Please send correspondence to:
Michael J. Buchenhorner, P.A.
1430 Sorolla Avenue
Coral Gables, Florida 33134

By: Michael J. Buchenhorner

Michael J. Buchenhorner
Registration No. 33,162
Telephone No.: (305) 529-0221

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the claims:

1. (Twice Amended) A method of managing memory in a multi-threaded processing environment including local thread stacks and local thread heaps, and a global heap, said method comprising:

creating an object in a thread heap; and

for a given thread, monitoring each object [that is local to the given thread] in its heap, to determine whether the object is accessible [from] by any thread other than the given thread.

6. (Twice Amended) The method as claimed in claim [3] 2 further comprising intercepting assignment operations to an object in the thread heap to determine whether the object status should be changed.

14. (Twice Amended) The method as claimed in claim 1 [further comprising assigning a status to certain objects that designates the objects as local objects] wherein certain objects are associated with a global status on creation thereof.

18. (Twice Amended) A system for managing memory in a multi-threaded processing environment comprising:

respective local thread stacks and heaps;

a global heap;

means for creating an object in a thread heap; and

means for monitoring each object [that is local to the given thread] in its heap, to determine whether the object is accessible [from] by any thread other than the given thread.

22. (Twice Amended) A computer program product stored on a computer readable storage medium for, when executed on a computer, performing a method of managing memory in a multi-threaded processing environment including local thread stacks and local thread heaps, and a global heap, said method comprising:

creating an object in a thread heap; and

for a given thread, monitoring each object [that is local to the thread] in its heap, to determine whether the object is accessible [from] by any thread other than the given thread.